Announcements

• Homework 6 due Dec. 10, last day of class

• Final exam Thursday, Dec. 17\textsuperscript{th}, 4-7 PM, Hamilton 602 (this room)
  • same format as midterm (open book/notes)

• Distinguished Lecture: Barbara Liskov, MIT. Turing Award Winner 2009.
  11 AM Monday. Davis Auditorium, CEPSR/Schapiro
Review

- External sorting: merge to alternating disks
- Complexity Classes
  - P, NP: Euler path
  - NP-Complete, NP-Hard: Hamiltonian path, Satisfiability, Graph Coloring
  - NP-Hard: Traveling Salesman
  - Undecidable: Halting Problem
Today’s Plan

• Note about hw4
• Finish discussion of complexity
  • Polynomial Time Approximation Schemes
  • Graph Isomorphism
• k-d trees
Rehashing

- A hash table does not store the input order
- When rehashing, elements are inserted into the new table in the array order
- No penalty on homework, but make sure it’s correct on the final
NP-Hardness

1. An algorithm for an NP-Hard problem can be used to solve any NP problem via polynomial time conversion.

2. But we don’t know algorithms to solve NP-Hard problems in poly. time.

3. If we did, P = NP, so most conjecture that NP-Hard problems must be intractable.
Poly. Time Approximation

- Certain optimization NP-Hard problems have polynomial time approximation schemes (PTAS)
- An efficient method to find a solution within a constant of the true optimum
  - e.g., Optimal TSP path length = $\ell$
  - PTAS TSP path length $\leq \ell(1 + \epsilon)$
- For fixed constant, must be poly. time, but can scale poorly w.r.t. constant
- E.g., $O(p(N)^{(1/\epsilon)!})$ is a valid PTAS time
The Graph Isomorphism problem is NP,
but is unknown if NP-Complete/Hard,
and no poly. time algorithm is known.
Graph Isomorphism
Definition

- Given graphs G and H, is there a 1-to-1 mapping of vertices from G to vertices from H that preserves the edge structure?

- Subgraph Isomorphism: is a subgraph of G isomorphic to H?
Complexity

P
- Euler Path,
  - Sorting,
  - Selection,
  - lots of stuff

NP
- Satisfiability,
- Hamiltonian Path,
- Subgraph Iso,
- Graph Coloring,
  - etc.

NP-Complete
- Graph Iso?

NP-Hard
- Traveling Salesman,
- Halting Problem
Advanced Data Structures

- We’ve mostly studied fundamental data structures
  - wide application areas, very general
- In practice, you’ll often have more specific goals, and thus need to design your own data structures
kd-Trees

- Useful data structure for data mining and machine learning applications
- Store elements by k-dimensional keys
  - e.g., age, height, weight
- Retrieve elements by ranges in the k dimensions
  - e.g., Searching for a new basketball center 18-24 year olds, 6’6”-7’4”, 200+ lbs
1-d Range Search

• BST recursive search:
  (1) if key is in range, print node
  (2) if key > lower bound, search left
  (3) if key < upper bound, search right

• $O(M + \log N)$ for $M$ items returned
• $O(\log N)$ to find nodes in range
• $O(1)$ at each node

Search for 3-7
kd-Tree Structure

- Binary search tree
- Each level splits on alternating keys
Search Algorithm

- Given lower and upper bounds for each dimension
- If key is in range, print
  If key > current dimension’s lower bound
    search left child
  If key < current dimension’s upper bound
    search right child
- Insert recursion is just like BST
2-d Tree
kd-Tree Analysis

- Since each level represents a different keys, balancing is not possible
- If we have all the points, we can build a perfectly balanced tree. How?
- Then worst case $O(M + kN^{1-1/k})$
Square Root vs. Log

The graph compares the square root function (red line) and the logarithmic function (blue line) for values of N ranging from 0 to 1000. The square root function grows relatively slowly compared to the logarithmic function, which increases at a much faster rate. The logarithmic function is represented by the blue line labeled 'log', while the square root function is represented by the red line labeled 'sqrt'.
Nearest Neighbor Search

- kd-trees are especially helpful for finding nearest neighbors
- Given a data set, find the nearest point to any element $x$
- Naive $O(N)$ approach is to compute distances everywhere
- Instead, kd-tree offers $O(kN^{1-1/k})$
Nearest Neighbor Algorithm

- Search for $x$ in the kd-tree until you reach a leaf
- Consider leaf point $current\text{-}best$
- Backtrack along search path, and at each node:
  - If current point is better, redefine $current\text{-}best$
  - If best can be in the unexplored child*, recurse down the unexplored child
Algorithm Illustration
kd-Tree Summary

- Smart data structure for storing and searching multi-dimensional keys
- Branch BST cycling dimensions at each level
- Running time is sub-linear, but grows to linear when k is infinite
- Efficient range search, search, and nearest-neighbor search
Reading

- Complexity - Weiss 9.7
- kd-trees - Weiss 12.6